Journal of
Low Power Electronics
Vol. 3, 1–10, 2007

A Low Power Design for Sbox Cryptographic Primitive of Advanced Encryption Standard for Mobile End-Users

George N. Selimis*, Athanasios P. Kakarountas, Apostolos P. Fournaris, Athanasios Milidonis, and Odysseas Koufopavlou

Department of Electrical and Computer Engineering, University of Patras, Greece, 26110

(Received: 16 June 2007; Accepted: 15 October 2007)

Most known Sbox Advanced Encryption Standard implementations aim at minimizing chip covered area or achieving high throughput, and usually power consumption is a secondary metric of their cost. However, the need for low power applications with strict chip covered area constrains is great especially in mobile devices. In this paper low power architectures in limited area resources are proposed for Sbox, the basic cryptographic primitive of AES. Those architectures support encryption and decryption operation modes. In our proposed implementations, retaining a small chip covered area cost, hardware techniques for low power design, such as re-ordering of the components, introduction of redundant hardware (ideal delay line), reducing the driving strength – fan out, and insertion of registers in highly unbalanced points of the circuit are applied. Therefore, our implementations do not only reduce power consumption by a large degree, but they also have good area properties and offer an advantageous power-delay-area product in comparison with other known Sbox implementations. These properties give to our system the advantage to support low power mobile devices in low area system environments.

Keywords: Advanced Encryption Standard, Sbox Transformation, Cryptography, VLSI, Hardware, Wireless Networks, Low-end Devices.

1. INTRODUCTION

In January 1997, the National Institute of Standards and Technology (NIST) invited new algorithm proposals for the Advanced Encryption Standard (AES) in order to replace the old Data Encryption Standard (DES). After two rounds of evaluation on the 15 candidate algorithms, NIST selected the Rijndael as the AES algorithm in October 2000.[1] Since NIST adopted AES algorithm as basic standard for symmetric cryptography, AES has become the main symmetric algorithm in many communication protocols and applications. The AES algorithm has broad applications, including mobile phones, cellular phones, smart cards, RFID tags, WWW servers and automated teller machines (ATMs).

The more widely used wireless protocols[2–3] adopted AES algorithm as their basic security mechanism[4] for providing system authentication, authorization and data integrity. However, in such systems, the constrains in power dissipation and chip covered area are very strict. Recent mobile devices (laptops, PDAs, etc.) can operate just for few hours before the battery gets exhausted. Even worse, the difference between power requirements of electronic components and battery capacities is expected to increase in the near future.[5–6] New passive devices as contactless smarts cards and RFID tags[7] that extract power from electromagnetic fields make the problem more complicated. Hardware design is known to accelerate algorithmic processes and decrease the requirements in power. Therefore, the existence of supplementary hardware is essential[8] if the designer's goal is to construct power efficient systems in limited area resources, like mobile systems.

The existing implementations of AES do not focus on the problem of power consumption but rather to present high throughput architectures.[9–10] Such architectures do not reach their full potential in mobile systems since such systems do not include high throughput specifications. In this paper we propose several hardware implementations for AES S-box primitive, that consists the main cryptographic primitive and the main factor of power consumption in the AES algorithm. In the works[11–12] it is proven that the power dissipation of Sbox cryptographic primitive is about the 75 percent of the AES overall power dissipation. So improving the power characteristics

*Author to whom correspondence should be addressed.
Email: gselimis@ece.upatras.gr

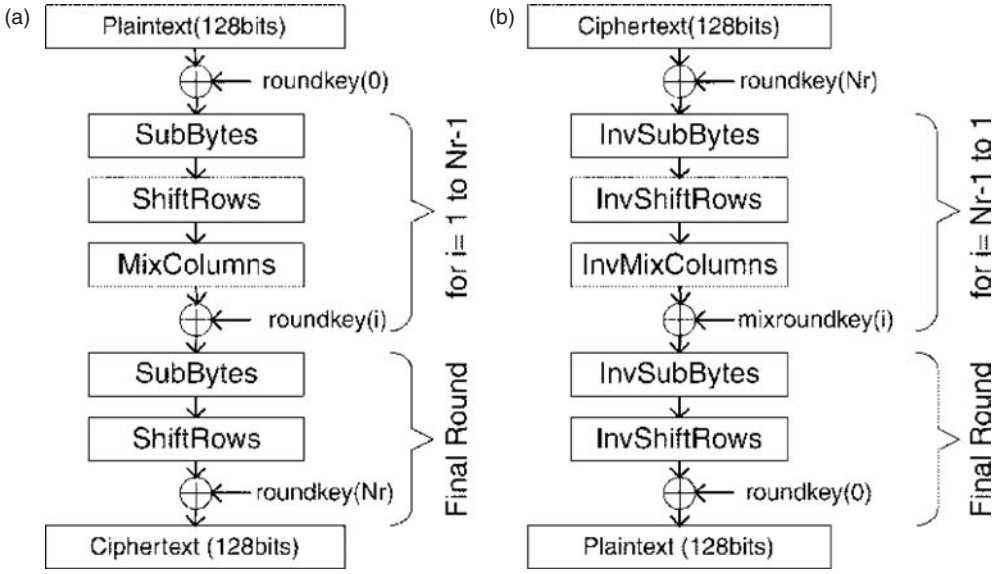

Fig. 1. Encryption and decryption processes of AES Algorithm.

of Sbox component we improve the algorithm power efficiency significantly. There exist many works[13–16] that try to give a more simplified architecture of AES Sbox in terms of chip covered area resources, ignoring that area reduction is not proportional to power reduction. However in mobile–wireless devices power consumption is the main constrain that have to be reduced. Achieving lower power consumption increases the battery efficiency of the system and it gives the opportunity for building more complex systems. In this paper we propose several Sbox implementations and we apply VLSI techniques to such systems achieving low power operation in limited area resources.

A brief basic structure of the standard AES is given in Section 2. In Section 3 the S-box transformation is presented in detail. In Section 4 the most significant approaches in S-Box design are reported. The proposed architecture is presented in Section 5. Experimental results and Comparisons are presented in Section 6. Finally, the paper is concluded in Section 7.

2. AES ALGORITHM

The AES algorithm is a symmetric-key cipher, in which both the sender and the receiver use a single key for encryption and decryption. The data block length is fixed to be 128 bits, while the key length can be 128, 192, or 256 bits, respectively. In addition, the AES algorithm is an iterative algorithm. Each iteration can be called a round, and the total number of rounds, N_r is 10, 12, or 14, when the key length is 128, 192, or 256 bits, respectively. The 128-bit data block is divided into 16 bytes. These bytes are mapped to a 4×4 array called the State, and all the internal operations of the AES algorithm are performed on the State. Each byte in the State is denoted by $S_{i,j}$ $(0 \le i, j < 4)$, and is considered as an element of $GF(2^8)$. Although different irreducible polynomials can be used to construct, $GF(2^8)$ field, the irreducible polynomial used in the AES algorithm is $p(x) = x^8 + x^4 + x^3 + x + 1$. Figure 1 shows the block diagram of the AES encryption and the equivalent decryption structures.

In the encryption of the AES algorithm, each round except the final round consists of four transformations: the SubBytes, the ShiftRows, the MixColumns, and the AddRoundKey, while the final round does not have the MixColumns transformation.

- SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
- ShiftRows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
- MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column using a linear transformation.
- AddRoundKey—each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Fig. 2. The effect of the SubBytes() transformation on the State.

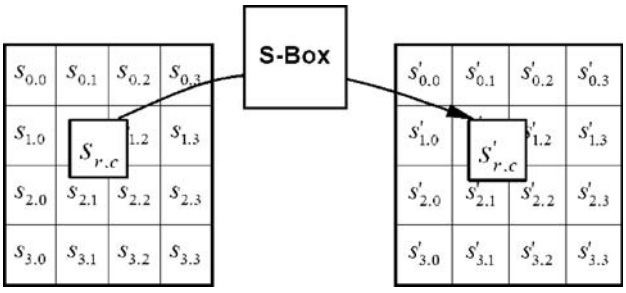

Fig. 3. SubBytes() applies the S-box to each byte of the State.

The previous Cipher transformations can be inverted and then implemented in reverse order to produce a straightforward Inverse Cipher for the AES algorithm. The individual transformations used in the Inverse Cipher are InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey.

3. SUBBYTES TRANSFORMATION

The SubBytes() (Ref. [1]) transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box (Fig. 4), which is invertible, is constructed by composing two transformations:

(1) Take the multiplicative inverse in the finite field $GF(2^8)$ of the input, the element {00} is mapped to itself.

(2) Apply the following affine transformation (over $GF(2)$):

$$b_i' = b_i \oplus b_{(i+4)} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i, \quad \text{for } 0 \le i \le 8 \quad (1)$$

where b_i is the i_ith bit of the byte, and c_i is the i_ith bit of a byte c with value {63} or {01100011}. Here and elsewhere, a prime on a variable (e.g., b_i') indicates that the variable is to be updated with the value on the right. In matrix form, the affine transformation element of the S-box can be expressed as:

The S-box used in the SubBytes() transformation is presented in hexadecimal form in Figure 4. For example, if $S_{i,j} = \{53\}$, then the substitution value would be determined by the intersection of the row with index '5' and the column with index '3' in Figure 4. This would result in $S_{i,j}'$ having a value of {ed}.

InvSubBytes() is the inverse of the byte substitution transformation, in which the inverse Sbox is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in $GF(2^8)$.

4. PREVIOUS SBOX IMPLEMENTATIONS

The simplest implementation of SBox transformation is the integration in silicon of the values presented in Figures 4 and 5 using specific look up tables. In this approach, the mathematical background of SBox transformation is not utilized for optimizations in any way. The optimization process of such implementation is based exclusively on the synthesis tool and assorted used libraries. Therefore, the power consumption is also depended on the synthesizer libraries. Since the synthesizer can run a highly sophisticated optimization process, the resulting circuit has small critical path delay and low power consumption. However, the resulting area of a look up table SBox transformation circuit is very high compared to other similar designs (about three times higher).[11, 13, 14, 16]

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig. 4. S-box: substitution values for the byte xy (in hexadecimal format).

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Fig. 5. Inverse S-box: substitution values for the byte xy (in hexadecimal format).

A similar implementation approach can be followed by replacing the AES SBox transformation by a ROM. In this case, the critical path delay is very similar to a hardware look-up table approach, but the power consumption of ROM is about 2–3 orders of magnitude higher that this approach.

A third approach for implementing the AES S-box was proposed by Bertoni et al. in Ref. [15]. By using an intermediate one-hot encoding of the input, arbitrary logic functions (including cryptographic S-boxes) can be realized with minimal power consumption. Designing the Sbox transformation using this approach the system would consists of an 8 to 2^8 encoder, a 2^8 to 8 decoder and a appropriate mapping of the encoder's output to the decoder's input. The main drawback of this approach is that the resulting implementation employs large silicon area on chip. However, this implementation supports only the encryption mode. For decryption, an identical system is required. For this reason the resulting implementation of the above system employs large silicon area on chip. Those area resources are considerably higher than the conventional look up table approach and the arithmetic implementation approaches are presented later. An additional disadvantage of Bertoni's implementation is that the architecture can not be easily modified and is not flexible. In order to make this architecture faster, a considerable amount of latches should be inserted in the data flow of the system thus increasing the chip covered area dramatically. This problem makes the above implementation not useful for mobile users devices with limited area resources.

There are more advanced approaches in designing Sbox transformation. Those approaches take advantage of the $GF(2^8)$ field arithmetic properties (arithmetic implementation approaches). The main issue in such implementations is the efficient realization of the inversion in $GF(2^8)$ that can be achieved by decomposing the finite field into the sub-fields $GF(2^4)$ and $GF(2^2)$. An inversion in a finite field of characteristic 2 can be carried out in various ways according to the employed element representation basis of such a field. The main advantage of these implementations is their small chip covered area cost because both encryption and decryption employ the same silicon core. However, the reduction in chip covered area does not lead to a corresponding reduction in power dissipation. On the contrary, in those arithmetic approaches the power dissipation is not considered. Due to possible glitches in the internal component and in the overall circuit topology (network), power dissipation can be 10–30 times higher than the lookup table approach.[17]

In this paper, we propose a methodology that can be used to synthesize cryptographic Sboxes on custom silicon (ASIC) libraries with energy-efficiency as a primary goal. For comparison reasons we implement look up table S-Box along with a representative arithmetic implementation by Wolkerstorfer et al.[13] and we proposed four optimized implementations of Ref. [13].

5. PROPOSED IMPLEMENTATION

As described in Section 4, the basic advantage of arithmetic implementations of S-BOX cryptographic primitive is their limited covered area resources. This fact is based on ability of those implementations to support both encryption and decryption using the same logic circuit. As already analyzed, the main problem of arithmetic implementations is the increased power dissipation in contrast to other implementations. We propose a new implementation approach that combines low power dissipation and low chip covered area for application with limited

resources. The proposed AES implementation is based on a modified S-Box[13] that can perform both encryption and decryption and the arithmetic of the $GF(2^8)$ finite field. The aim of this work is to provide an area cost-effective solution of an AES implementation with low-power constrains. This means that power dissipation is one of the main design parameters and cost (i.e., redundant hardware) can be altered for succeeding in this. To the best of our knowledge, there has never been proposed such an implementation that combines low power dissipation with low area specifications. Four alternative implementations are described in detail named Implem1, Implem2, Implem1R, and Implem2R. All the steps of the optimization process for the architectures Implem1 and Implem2 are presented. Implem1 and Implem2 have their corresponding pipelined versions Implem1R and Implem2R.

The proposed S-Box can be modified to present low-power dissipation, for either full-custom or semi-custom technologies, by balancing the delays of the logic gates' inputs and improving several characteristics of the circuit. This can be achieved by using techniques, such as re-ordering of the components, introduction of redundant hardware (ideal delay line), and by reducing the driving strength – fan out, and insertion of registers in highly unbalanced points of the circuit. One of the main dissipation sources is the dynamic power dissipation caused by the consumption of power either on the logic gates or on the wires that compose the circuit's network. The first parameter (power dissipation on gates) can be minimized if the inputs are applied simultaneously and all the transitions in all the inputs of the gate occur at the same time instance. This is an ideal and unfortunately only theoretical scenario, in which power is consumed solely for the calculation of the gate's output. We refer to this scenario as 'zero-delay' model.

The second parameter (power dissipation on the network) can be minimized only by controlling the outputs of the logic gates and hence eliminate spurious and desirable transitions. The so called signal glitching not only causes power consumption on the nets but furthermore triggers the logic gates at their end, forcing them to calculate a faulty output.

Some common techniques to address the aforementioned problems are either the use of custom sizes of transistors, with a variety of voltage thresholds and noise margins, or the modification of the circuit's logic function. The first approach is trivial for every circuit design, it is applicable for high-speed circuits targeting mainly technologies allowing full-custom design. The second approach can be easily applied on various technologies and it is based mainly on the functionality of the circuit and not its physical implementation. The higher power saving in a vast number of VLSI implementations is based on this high-level design approach. This fact, makes it appropriate for our goal.

In the following paragraphs our proposed methodology is discussed in detail. The implementation is based in finite fields operations thus it employs appropriate components for multiplication, squaring, inversion, XOR operation and some extra logic in order to enable encryption/decryption operation modes. In Figure 6 the logic that creates main signal highlighted. The contribution of those glitches is significant in the overall power dissipation.

The network's delays (shaded network in Fig. 6) are balanced by inserting an extra XOR gates level. In Figure 7(a) (Implem1 architecture), the result of such operation is the creation of the COMP component. This approach contributes significantly to the elimination of spurious transitions that were caused due to the misdistribution of propagation delays. The resulted structure of the SBox seems more balanced as far as it concerns the glitch generation of the INVERSION component's inputs. However, this statement is not true since components COMP and MULGF(2^4) have different critical paths. Thus, additional steps to further optimize the balancing of the propagated signal delays on the main data path, are the flattening of component COMP, the analysis of its functionality, its optimization in terms of speed and finally, the creation of an output datapath with a delay similar to that of the component MULGF(2^4).

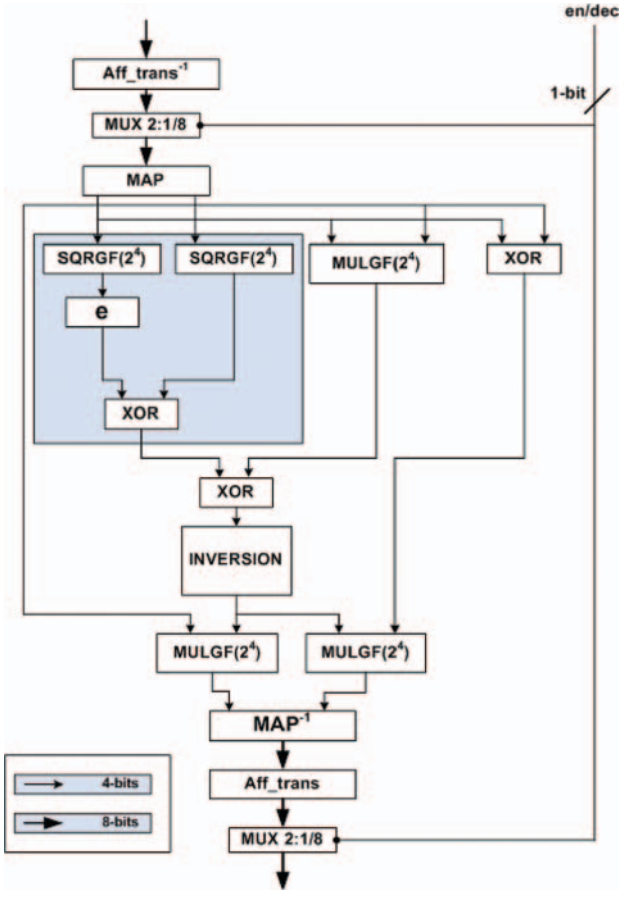

Fig. 6. The conventional SBox architecture.

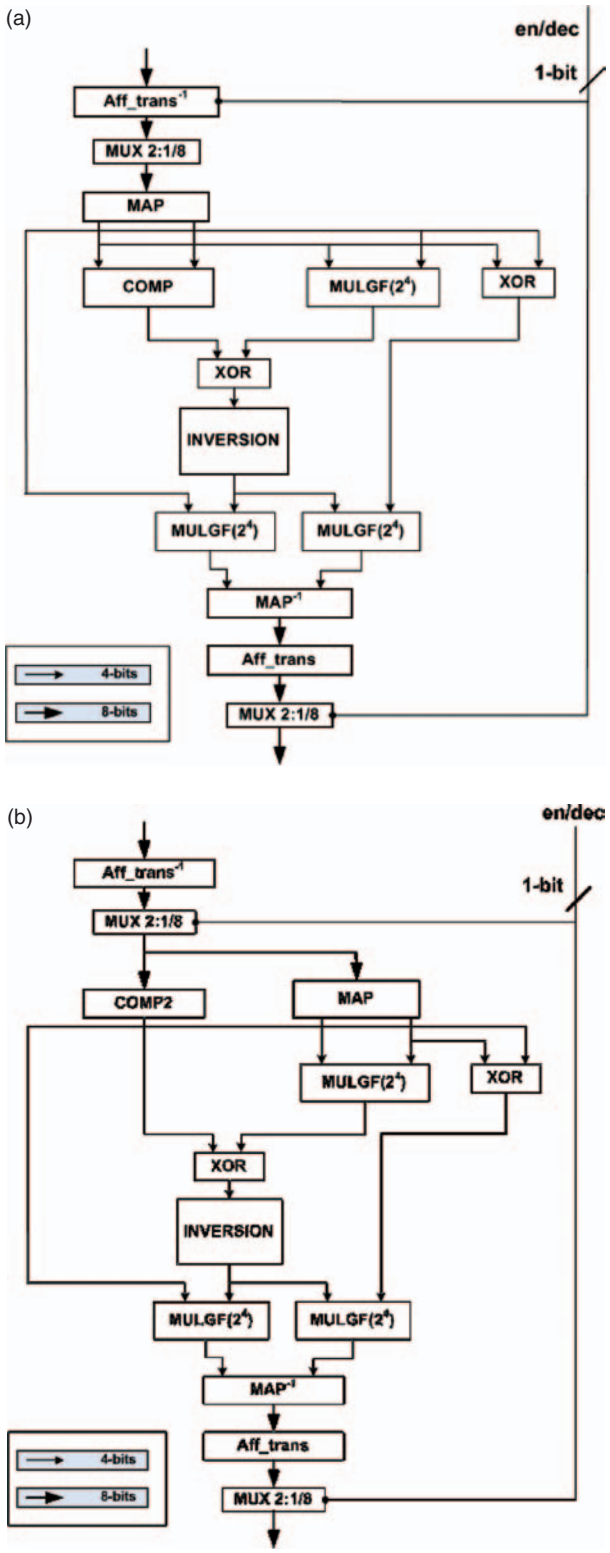

Fig. 7. The modified SBox architectures (a) Implem1 and (b) Implem2.

Flattening of the COMP circuit reveals that it consists only from XOR gates, which is ideal for creating a balanced XOR tree (Fig. 7(a) – Implem1). Analysis of COMP component's functionality and having in mind that only XOR logic gates are utilized, reveals that all the output bits can be expressed solely as a function of the input bits, without considering any intermediate values. The optimization of the XOR tree, consists of two steps. Firstly, the logic is simplified using simple Boolean Algebra. The result was quite interesting since it was discovered that porting the SBox in finite fields, eliminates several dependencies. Thus, the complex initial logic can be substituted by small XOR trees. Secondly, extra XOR logic gates were introduced in order to balance the delays throughout the XOR trees. The same procedure (insertion of XOR logic gates) is followed in order to equalize the output delays of the components COMP and MULGF(2^4).

A further optimization (Implem2) for decreasing the power dissipation is to modify several characteristics of the circuit. A critical issue of the initial implementation was the high fan-out of the component MAP. The use of buffers to form the appropriate driver (superbuffer), increases not only the cost but also the propagation delay. Thus the optimization can be focused on reducing the fan-out of the component MAP. This can be achieved with two different approaches. The first approach is to duplicate the component MAP and share the fan-out to the two replicas. Although this is straightforward, the tradeoff between a superbuffer and two MAP components is not fair, in terms of cost. The other approach, which is not always applicable, is to transform the component COMP to have as inputs the inputs of component MAP, by correlating the two inputs and discover their dependencies.

The result of the latter optimization is illustrated in Figure 7(b) (Implem2). In order to make clearer how the component COMP resulted in its flattened small sized implementation, the series of simplifications, using Boole Algebra (the known property that $a(i) \oplus a(i) = 0$), is offered below. In Figure 8, the nodes of the circuit that can be optimized are numbered and the analysis and simplification process that is followed is presented below:

Node 2

$$\begin{aligned} ah(0) &= a(5) \oplus a(4) \oplus a(6) \\ ah(1) &= a(1) \oplus a(7) \oplus a(4) \oplus a(6) \\ ah(2) &= a(5) \oplus a(7) \oplus a(2) \oplus a(3) \\ ah(3) &= a(5) \oplus a(7) \end{aligned} \tag{2}$$

Node 3

$$\begin{aligned} sq(0) &= ah(0) \oplus ah(2) \\ &= a(5) \oplus a(4) \oplus a(6) \oplus a(5) \oplus a(7) \oplus a(2) \oplus a(3) \\ &= a(4) \oplus a(6) \oplus a(7) \oplus a(2) \oplus a(3) \\ sq(1) &= ah(2) = a(5) \oplus a(7) \oplus a(2) \oplus a(3) \\ sq(2) &= ah(1) \oplus ah(3) \\ &= a(1) \oplus a(7) \oplus a(4) \oplus a(6) \oplus a(5) \oplus a(7) \\ &= a(1) \oplus a(4) \oplus a(5) \oplus a(6) \\ sq(3) &= ah(3) = a(5) \oplus a(7) \end{aligned} \tag{3}$$

Node 4

$$\begin{aligned}
e(0) &= sq(1)\oplus sq(2)\oplus sq(3)\\
&= a(5)\oplus a(7)\oplus a(2)\oplus a(3)\\
&\quad \oplus a(1)\oplus a(4)\oplus a(5)\oplus a(6)\oplus a(5)\oplus a(7)\\
&= a(2)\oplus a(3)\oplus a(4)\oplus a(5)\oplus a(6)\oplus a(1)\\
e(1) &= sq(0)\oplus sq(1)\\
&= a(4)\oplus a(6)\oplus a(7)\oplus a(2)\oplus a(3)\\
&\quad \oplus a(5)\oplus a(7)\oplus a(2)\oplus a(3)\\
&= a(4)\oplus a(5)\oplus a(6)\\
e(2) &= sq(0)\oplus sq(1)\oplus sq(2)\\
&= a(4)\oplus a(6)\oplus a(7)\\
&\quad \oplus a(2)\oplus a(3)\oplus a(5)\oplus a(7)\oplus a(2)\oplus a(3)\\
&\quad \oplus a(1)\oplus a(4)\oplus a(5)\oplus a(6) = a(1)\\
e(3) &= sq(0)\oplus sq(1)\oplus sq(2)\oplus sq(3)\\
&= a(4)\oplus a(6)\oplus a(7)\\
&\quad \oplus a(2)\oplus a(3)\oplus a(5)\oplus a(7)\oplus a(2)\oplus a(3)\\
&\quad \oplus a(1)\oplus a(4)\oplus a(5)\oplus a(6)\oplus a(5)\oplus a(7)\\
&= a(1)\oplus a(5)\oplus a(7)
\end{aligned} \tag{4}$$

Node 5

$$\begin{aligned}
al(0) &= a(4)\oplus a(6)\oplus a(0)\oplus a(5)\\
al(1) &= a(1)\oplus a(2)\\
al(2) &= a(1)\oplus a(7)\\
al(3) &= a(2)\oplus a(4)
\end{aligned} \tag{5}$$

Node 6

$$\begin{aligned}
sq'(0) &= al(0)\oplus al(2)\\
&= a(4)\oplus a(6)\oplus a(0)\oplus a(5)\oplus a(1)\oplus a(7)\\
&= a(0)\oplus a(1)\oplus a(4)\oplus a(5)\oplus a(6)\oplus a(7)\\
sq'(1) &= al(2) = a(1)\oplus a(7)\\
sq'(2) &= al(1)\oplus al(3)\\
&= a(1)\oplus a(2)\oplus a(2)\oplus a(4) = a(1)\oplus a(4)\\
sq'(3) &= al(3) = a(2)\oplus a(4)
\end{aligned} \tag{6}$$

Node 7

$$\begin{aligned}
x(0) &= e(0)\oplus sq'(0) = a(0)\oplus a(2)\oplus a(3)\oplus a(7)\\
x(1) &= e(1)\oplus sq'(1) = a(1)\oplus a(7)\oplus a(4)\oplus a(5)\oplus a(6)\\
x(2) &= e(2)\oplus sq'(2) = a(4)\\
x(3) &= e(3)\oplus sq'(3) = a(2)\oplus a(4)\oplus a(1)\oplus a(5)\oplus a(7)
\end{aligned} \tag{7}$$

In our application, the latter optimization approach can be effective. Initially, there was a fun-out problem of the component MAP. However, the simplification of the functions in COMP results in a modified component COMP2, which is driven by the inputs of the component MAP, instead of its outputs. This is critical for the size of the transistors driven by the outputs of the MAP component as the requirements for a good driving strength are now relaxed. Furthermore, the outputs of the modified component are responding very fast, offering a bigger degree of freedom to alter the propagation delay, of the COMP2 signals, appropriately. This results in balancing the propagation delay of the whole circuitry, thus maximizing the power savings when compared to non-pipelined implementations (lookup tables, arithmetic implementations).

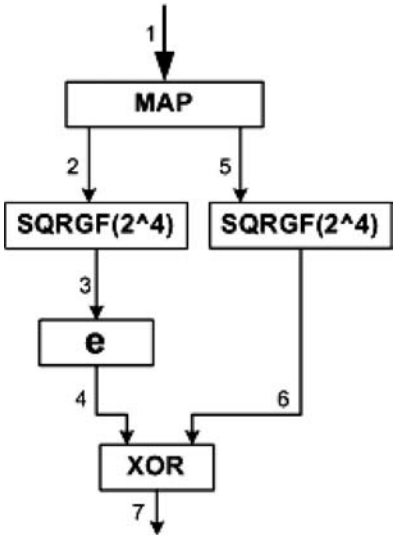

Fig. 8. Circuit that can be simplified.

The implementations implem1R and implem2R are realized at the final optimization step for increasing the power savings even more, is to introduce several pipeline stages throughout the circuit. The insertion of pipeline registers eliminates, at the stage of insertion, the propagation of glitches. This is a useful technique in order to completely control power dissipation, paying however the appropriate cost. Considering that the latter optimization steps have reduced area requirements in terms of logic, driving instances and transistor sizes, it is fair to agree that the tradeoff between cost and power decrease satisfies in total the initial application constrains (mobile-end user).

The power savings when inserting the pipeline registers at the appropriate location is expected to be at least 2 times higher than those achieved with non-pipelined implementation. The estimation is fair for circuitry that cannot be easily balanced in terms of propagation delay. However, in our application, that the optimization steps have balanced significantly the propagation delay of the signals throughout the circuitry, the power saving is expected to be much higher for a fair penalty in cost. As it will be shown below, where the experimental results of our implementations are presented, the overall power dissipation is one degree of magnitude lower than that of the non-pipelined version.

6. COMPARISONS AND EXPERIMENTAL RESULTS

All implementations have been synthesized with Synopsys synthesis tools (Design Compiler) on a Linux platform

and the target technology library is 0.18 μm with 1.8 V core voltage. All post-synthesis simulations and generation of the switching activity information have been done at the logic gate level with 1 ps resolution, using the VHDL simulator Modelsim of Mentor Graphics. The total power dissipation is measured using Synopsys' PrimePower.

We implement six SBOX approaches in VHDL. The lookup table approach and Wolkerstorfer approach are implemented for comparison reasons. The other four implementations consist four distinct optimization steps that we can follow in order to propose a final optimized approach. In Figures 10–13 the main problems of the arithmetic implementation of Wolkerstorfer[13] are highlighted. Those problems are the high power dissipation and critical path delay. These parameters make the area optimization of Ref. [13] useless since the trade-off in power and delay is unacceptable. Furthermore, as shown in Figures 10, 11, although the power dissipation table implementation, the area resources are very high compared to all the other implementations. Figures 10–13 also present implementation results for the proposed four implementations. Without the use of any register, very satisfactory results are achieved for implementation2 (implem2). The Power × Area × Delay, (PAD) product, that offers a fair measure of the achieved optimization degree, is about the same as the conventional lookup table implementation by achieving much smaller chip covered area and slightly more power dissipation. The use of registers, as is introduced in implementations implem1R, implem2R, give better results by one order of magnitude in comparison to all the other implementations with the exception of the chip covered area. The implementation results are shown analytically in Table I. From this table, it can be concluded that the proposed architecture fitting best in accordance to our goals

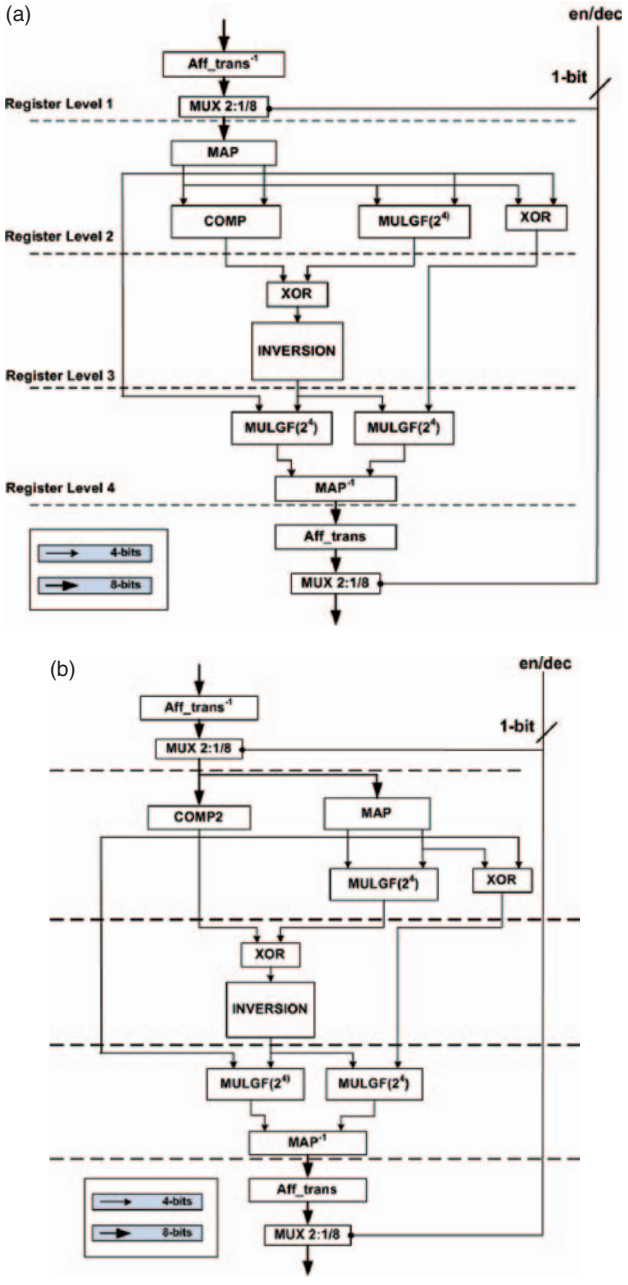

Fig. 9. The (a) Implem2 and (b) Implem3 SBox architectures with registers.

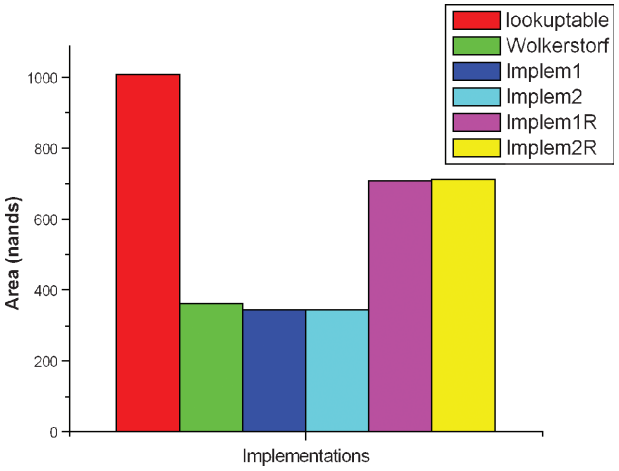

Fig. 10. Area comparisons of AES SBox architectures.

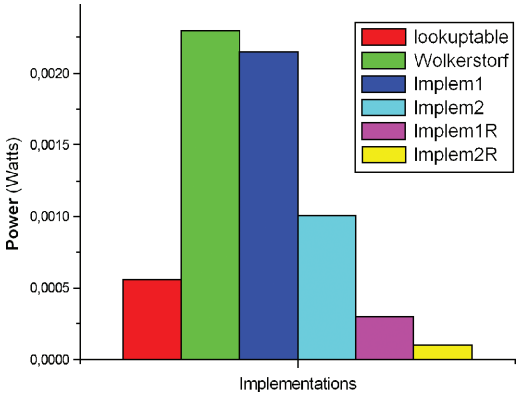

Fig. 11. Power comparisons of AES SBox architectures.

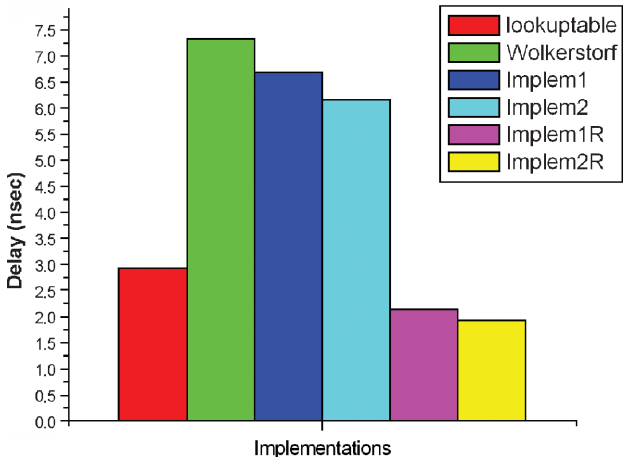

Fig. 12. Delay comparisons of AES SBox architectures.

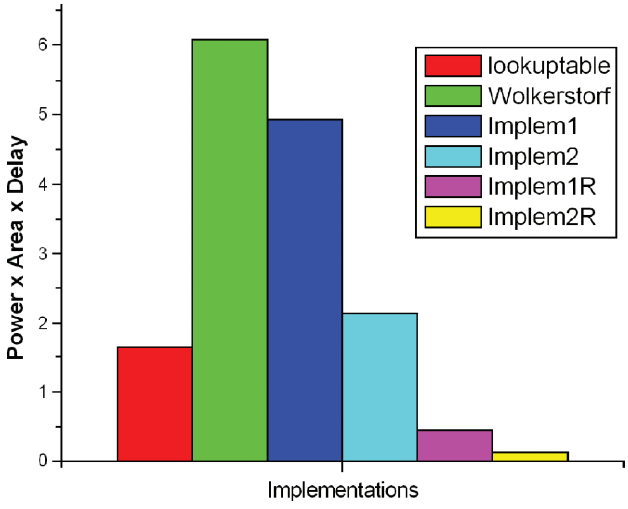

Fig. 13. Power × Area × Delay product comparisons of AES SBox architectures.

Table I. Implementation comparisons of AES SBox architectures.

	Power (W)	Area (cells)	Delay (ns)	P × A product	P × A × D product
Lookup tables	0.00056	1009.2384	2.93	0.56504	1.655958366
Wolkerstorfer[13]	0.0023	362.2191	7.33	0.8331	6.106651807
Implem1	0.00215	343.84275	6.69	0.7392	4.94556622
Implem2	0.001007	344.5444	6.17	0.3469	2.140719821
Implem1R	0.0003	708.977	2.15	0.2126	0.45729017
Implem2R	0.0001	712.6414	1.93	0.0712	0.1375398

of low power and low area constrains with out using registers, is implementation2 (implem2). The Power × Area (PD) product, that offers a fair measure of the achieved optimization degree following the low area-low power constrains, has the lowest value in the no register cases when implementation 2 is used.

7. CONCLUSION

In this paper several implementations of AES SBox transformations were proposed. Using arithmetic implementation logic for achieving low chip covered area resources we enhance our implementations with low power techniques in order to reduce the power dissipation of such implementations. Comparing the proposed implementations with other similar works, confirm the benefits of the proposed designs in low power, low chip covered area systems. Such implementations can be used efficiently in mobile devices where the need for low power dissipation and small chip covered area is great.

References

1. National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November **(2001)**.
2. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification **(1999)**.
3. IEEE Std. 802.16-2001, IEEE Standard for Local and Metropolitan Area Networks, part 16, Air Interface for Fixed Broadband Wireless Access Systems, IEEE Press **(2001)**.
4. IEEE Standard 802.11i-2004: Standard for Information technology—Telecommunication and information exchange between systems-Local and metropolitan area networks-Specific requirements, July **(2004)**.
5. Heinrich Meyr and Tilman Glokler, Design of Energy-Efficient Application-Specific Instruction Set Processors, Kluwer Academic Pub. **(2004)**.
6. A. Boukerche, Handbook of Algorithms for Wireless Networking and Mobile Computing, Chapman & Hall/CRC **(2005).**
7. K. Finkenzeller, RFID Handbook—Fundamentals and Applications in Contactless Smart Cards & Identification, John Wiley and Sons Ltd **(2003)**.
8. O. Koufopavlou, G. Selimis, N. Sklavos, and P. Kitsos, Cryptography: Circuits and systems approach. *5th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT05)*, Greece **(2005)**.
9. X. Zhang and K. K. Parhi, High-speed VLSI architectures for the AES algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12, 957 **(2004)**.
10. A. Hodjat and I. Verbauwhede, Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors. *IEEE Transactions on Computers* 55 **(2006)**.
11. S. Morioka and A. Satoh, An Optimized S-Box Circuit Architecture for Low Power AES Design, CHES **(2002)**.
12. A. Satoh, S. Morioka, K. Takano, and S. Munetoh, A Compact Rijndael Hardware Architecture with S-Box Optimization. ASIACRYPT **(2001)**.
13. J. Wolkerstorfer, E. Oswald, and M. Lamberger, An ASIC implementation of the AES SBoxes, Topics in Cryptology CT-RSA, Vol. 2271 of Lecture Notes in Computer Science, Springer Verlag **(2002)**.
14. P. Chodowiec and K. Gaj, Very compact FPGA implementation of the AES algorithm, Cryptographic Hardware and Embedded Systems—CHES 2003, Vol. 2779 of Lecture Notes in Computer Science, Springer Verlag **(2003)**, pp. 319–333.
15. G. Bertoni, M. Macchetti, L. Negri, and P. Fragneto, Power-efficient ASIC synthesis of cryptographic Sboxes. *Proceedings of the 14th ACM Great Lakes Symposium on VLSI (GLSVLSI 2004)*, ACM Press **(2004)**, pp. 277–281.

16. D. Canright, A very compact S-Box for AES, Cryptographic Hardware and Embedded Systems CHES 2005, Vol. 3659 of Lecture Notes in Computer Science, Springer Verlag **(2005)**, pp. 441–455.

17. S. Tillich, M. Feldhofer, and J. Großschädl, Area, delay, and power characteristics of standard-cell implementations of the AES S-Box, Embedded Computer Systems: Architectures, Modeling, and Simulation—SAMOS 2006, edited by Stamatis Vassiliadis, Stephan Wong, and Timo D. Hämäläinen, Vol. 4017 of Lecture Notes in Computer Science, Springer Verlag (**2006**), pp. 457–466.

18. G. Selimis, A. P. Fournaris, and O. Koufopavlou, Applying low power techniques in AES MixColumn/InvMixColumn transformations. *Proceedings of 13th IEEE International Conference on Electronics, Circuits and Systems (ICECS'06)*, Nice, France, December **(2006)**.

George N. Selimis

George N. Selimis *received his Diploma in Electrical and Computer Engineering from the Electrical and Computer Engineering Department, University of Patras, Greece in 2003. He started his Ph.D. in September 2003 in the same department. His research interests include Cryptographic Engineering, VLSI low power design, and Security in Wireless Networks. He has also participated as Senior Engineer in Greek and European Research Projects.*

Athanasios Kakarountas

Athanasios Kakarountas *received his Diploma in Engineering and Ph.D. degree from University of Patras, Greece. In 2004 he joined the VLSI Design Laboratory, ECE Department of the University of Patras as Senior Researcher. He holds also a visiting position at the University of Central Greece, Lamia since mid 2006. His research interests include the dependable computing field (both safety and security) and low-power system design. He is author and co-author of more than 50 research papers in international journals and conferences and he is the co-recipient of three student awards and two international student awards.*

Apostolos P. Fournaris

Apostolos P. Fournaris *was born in 1978 in Athens, Greece. He received his Diploma on Electrical and Computer Engineering after 5 years of study from the University of Patras in November 2001. For one year he has done extensive research in analog circuit design and especially PLL's. From September 2002 he's been working toward his Ph.D. degree in the department of Electrical and Computer Engineering on the subject of VLSI design for cryptographic applications and expecially Public key encryption algorithms. He has published several papers in international conferences and is a reviewer for many international conferences and journals. He has extensive teaching experience as a teaching assistant in several undergraduate laboratories.*

Athanasios Milidonis

Athanasios Milidonis *received a Diploma in Electrical and Computer Engineering in 2000, M.Sc. in Hardware and Software for Integrated Systems in 2002 and Ph.D. in 2007 from University of Patras in Greece. His research interests include the domains of Memory Management and Embedded Software and Hardware.*

Professor Odysseas Koufopavlou

Professor Odysseas Koufopavlou *received the Diploma of Electrical Engineering in 1983 and the Ph.D. degree in Electrical Engineering in 1990, both from University of Patras, Greece. From 1990 to 1994 he was at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He is currently Professor with the Department of Electrical and Computer Engineering, University of Patras. His research interests include computer networks, high performance communication subsystems architecture and implementation, VLSI low power design, and VLSI crypto systems. Dr. Koufopavlou has published more than 150 technical papers and received patents and inventions in these areas. He has participated as coordinator or partner in many Greek and European R&D programmes. He served as general chairman for the IEEE ICECS'1999.*